

AGT DISPLAYS

Atualização de Arquivos pela UART

Gravação na Flash de 16 MB pela interface de variável de sistema 0x00AA

INTERFACE	APLICAÇÃO
Variável de sistema 0x00AA	Módulos com ASIC DWIN rodando DGUS II
MEIO	PÚBLICO
UART, sem cartão SD	Engenheiros de firmware e de campo

1. O QUE A 0X00AA FAZ

A interface de variável de sistema **0x00AA** grava blocos de dados diretamente na Flash externa de 16 MB do módulo – a mesma memória em que o download por cartão SD deposita imagens, ícones, fontes e arquivos de configuração. Como toda variável de sistema, ela é acessada pelas instruções 0x82 (escrita) e 0x83 (leitura) da UART. É o caminho para atualizar arquivos de projeto em campo, com o equipamento montado e sem abrir o gabinete para chegar ao soquete de cartão.

A gravação acontece em **duas etapas**, e essa é a parte que costuma confundir. A 0x00AA não recebe o arquivo: ela move para a Flash um bloco que *já está* na RAM de variáveis. Quem coloca o arquivo na RAM é uma sequência de instruções 0x82 comuns, em pedaços de algumas dezenas de bytes. A 0x00AA entra só no fim de cada bloco, para consolidar o que foi acumulado.

FLUXO EM UMA LINHA

Arquivo no PC ou no MCU → fatias de 240 bytes por 0x82 → área de espera na RAM → 0x00AA consolida 32 KB na Flash → repete até acabar o arquivo → reset do módulo.

QUANDO USAR

- **Atualização remota ou em campo** de um arquivo específico: uma tela de fundo, um ícone, uma fonte, a imagem de abertura.
- **Equipamento lacrado**, em que o cartão SD não é acessível.
- **Atualização controlada pelo próprio MCU**, que recebe o arquivo por rede, USB ou pen drive e repassa ao display.

QUANDO NÃO USAR

- **Carga inicial do projeto completo.** O download por SD trabalha a 4 Mb/s; a UART, a cerca de 12 KByte/s. Um projeto de alguns megabytes levaria horas.
- **Atualização do kernel DGUS ou do programa 8051.** São outras interfaces – veja a seção 8.
- **Arquivos grandes sem necessidade.** Mesmo uma ICL de 4 MB exige cerca de 17 mil quadros seriais. Prefira atualizar apenas o arquivo que mudou.

2. ESTRUTURA DA INSTRUÇÃO

O endereço 0x00AA ocupa **6 words** — 12 bytes, nomeados D11 (byte mais significativo) a D0. D11 dispara a operação e D10 escolhe o modo; os demais campos mudam de significado conforme o modo.

CAMPO	TAM.	DEFINIÇÃO
D11	1 byte	0x5A = inicia uma operação de memória externa. O próprio módulo zera este byte ao concluir.
D10	1 byte	Modo de operação (tabela abaixo).
D9...D0	10 bytes	Parâmetros do modo. Campos não usados devem ser escritos com 0x00.

MODOS DISPONÍVEIS

D10	OPERAÇÃO	USO
0x01	Leitura da NOR Flash SPI de 16 MB	Conferir conteúdo gravado; ler dados de fonte para a RAM.
0x02	Escrita de bloco de 32 KB na NOR Flash	É o modo da atualização de arquivos de projeto. Todo o restante deste documento trata dele.
0x03	Cópia de fontes dentro da Flash	Duplicar um arquivo já gravado para outro ID, sem passar pela serial.
0x04	Escrita de blocos de 4 KB na NAND Flash	Só em módulos com NAND expandida.
0x05	CRC32 de dados na NAND	Verificação pós-gravação. Mesmo polinômio do CRC do download por SD.
0x06	Cópia interna na NAND	Retoma sozinha após queda de energia, até concluir.

CAMPOS DO MODO 0X02

CAMPO	CONTEÚDO	FAIXA
D11	0x5A — dispara a operação	fixo
D10	0x02 — escrever bloco de 32 KB	fixo
D9:D8	Endereço do bloco de 32 KB na Flash	0x0000–0x01FF (16 MB)
D7:D6	Endereço na RAM de variáveis onde o bloco foi montado. Deve ser par.	0x0000–0xFFFF
D5:D4	Atraso após a gravação, antes da próxima escrita, em ms	0x0000–0xFFFF
D3:D0	Não definidos	escrever 0x00

FAIXA DE D9:D8 – a definição do campo fala em 0x0000–0x01FF, que corresponde a uma Flash de 16 MB (512 blocos de 32 KB). Em módulos com Flash expandida a faixa acompanha o total instalado: quatro chips de 16 MB dão 2048 blocos, isto é, 0x0000–0x07FF. Confirme a memória do modelo antes de calcular endereços.

O campo D5:D4 merece atenção: durante esse atraso **o DGUS interrompe a atualização da tela**. É intencional – evita que um refresh pegue a Flash no meio da escrita e produza dado incompleto. Um valor da ordem de 20 ms (0x0014) é o usado no exemplo do manual.

3. OS TRÊS CÁLCULOS

Antes de enviar qualquer byte é preciso resolver três contas. Errar qualquer uma delas grava o arquivo no lugar errado – e o efeito prático disso é tela preta, porque escrever em cima de um espaço ocupado apaga o que estava lá.

3.1. BLOCO DE DESTINO NA FLASH

A Flash de 16 MB é vista de duas formas ao mesmo tempo: como **64 subespaços de 256 KB** – são os IDs de arquivo que você usa no DGUS, de 0 a 63 – e como **512 blocos de 32 KB**, que é a granularidade da 0x00AA. Converter de um para o outro:

```
bloco_inicial = ID_do_arquivo × 256 ÷ 32
```

Exemplo, arquivo 32.ICL: $32 \times 256 \div 32 = 256 = 0x0100$

Ou seja, o primeiro pacote do arquivo 32 vai para o bloco 0x0100. Cada bloco de 32 KB gravado em seguida incrementa esse valor em 1.

3.2. ÁREA DE ESPERA NA RAM

Um bloco de 32 KB ocupa 16384 words na RAM de variáveis, ou seja, 0x4000 endereços. Escolhendo 0x8000 como início, a área vai de 0x8000 a 0xC000:

```
32 × 1024 ÷ 2 = 16384 words = 0x4000
```

```
0x8000 + 0x4000 = 0xC000
```

CUIDADO COM A FAIXA ESCOLHIDA — essa área não pode ser usada para mais nada enquanto a atualização corre, e 0x8000–0xC000 é justamente a região que a recomendação da DWIN reserva para os descritores SP. Em um projeto que usa SP, escolha outra faixa livre de 0x4000 words, ou aceite que os atributos SP sejam reescritos e recarregue-os depois.

3.3. FATIAMENTO EM QUADROS SERIAIS

Um quadro DGUS não passa de 255 bytes, o que deixa no máximo 249 bytes de dados. O manual adota **240 bytes** por quadro, que é o número que fecha as contas com facilidade:

GRANDEZA	CÁLCULO	RESULTADO
Quadros por bloco de 32 KB	$32 \times 1024 \div 240$	137 quadros (o último, parcial)
Quadros de um arquivo de 4,17 KB	$4,17 \times 1024 \div 240$	18 quadros
Passo de endereço entre quadros	$240 \div 2$	0x78 words

O passo é em *words* porque o endereçamento da RAM de variáveis é por word: 240 bytes equivalem a 120 endereços. Partindo de 0x8000, os inícios são 0x8000, 0x8078, 0x80F0, 0x8168, e assim por diante.

4. PROCEDIMENTO

A sequência abaixo atualiza um arquivo. Para arquivos maiores que 32 KB, os passos 2 e 3 se repetem bloco a bloco, incrementando D9:D8 a cada volta.

1. **Prepare o arquivo.** Gere o ICL, a fonte ou o arquivo de configuração no DGUS exatamente como faria para o cartão SD, e abra-o em um editor hexadecimal. É dele que saem os bytes dos quadros.
2. **Carregue o bloco na RAM.** Envie os quadros 0x82 de 240 bytes, o primeiro no início da área de espera e cada um seguinte 0x78 words adiante. No último quadro, ajuste o campo de comprimento ao número real de bytes – este é o erro mais comum de todos.
3. **Dispare a 0x00AA.** Um único quadro 0x82 escrevendo as 6 words em 0x00AA, com o bloco de destino e o endereço da área de espera.
4. **Aguarde o fim da gravação.** Leia 0x00AA com 0x83 e espere o byte de comando zerar (seção 5). Não envie o bloco seguinte antes disso.
5. **Repita** para os demais blocos, se houver.
6. **Reinicie o módulo.** A tela só passa a usar o arquivo novo depois do reset.

5. EXEMPLO COMPLETO – ATUALIZAR 32.ICL

Este é o exemplo do manual, reproduzido passo a passo: um arquivo 32.ICL de 4,17 KB, área de espera em 0x8000, módulo com Flash padrão de 16 MB. Como o arquivo cabe em um único bloco de 32 KB, há apenas uma chamada da 0x00AA.

5.1. CONTAS DO CASO

```
Bloco de destino ..... 32 × 256 ÷ 32 = 0x0100
Área de espera ..... 0x8000 (ocupa até 0xC000)
Quadros ..... 4,17 KB ÷ 240 B = 18 quadros
Passo ..... 0x78 words
Endereços ..... 0x8000 0x8078 0x80F0 0x8168 0x81E0 0x8258
                  0x82D0 0x8348 0x83C0 0x8438 0x84B0 0x8528
                  0x85A0 0x8618 0x8690 0x8708 0x8780 0x87F8
```

5.2. PRIMEIRO QUADRO DE DADOS

```
5A A5 F3 82 80 00 44 47 55 53 5F 33 FB 29 00 00 ...
```

BYTES	SIGNIFICADO
5A A5	Cabeçalho do quadro
F3	Comprimento: 243 bytes = instrução (1) + endereço (2) + dados (240)
82	Escrita em memória de variáveis
80 00	Endereço de destino na RAM – início da área de espera
44 47 55 53 ...	240 bytes do arquivo. Aqui, a assinatura DGUS_3 do cabeçalho do ICL

5.3. ÚLTIMO QUADRO

O décimo oitavo quadro carrega apenas 182 bytes, e por isso o campo de comprimento muda de 0xF3 para 0xB9 – 185 bytes:

```
5A A5 B9 82 87 F8 51 45 14 00 ... 51 45 14 01 FF D9
```

Os dois últimos bytes, FF D9, são o fim de imagem JPEG – confirmação visual de que o arquivo foi transmitido inteiro.

5.4. COMANDO DE GRAVAÇÃO

```
5A A5 0F 82 00 AA 5A 02 01 00 80 00 00 14 00 00 00 00
```

BYTES	CAMPO	SIGNIFICADO
5A A5	—	Cabeçalho
0F	—	15 bytes: instrução + endereço + 12 bytes de dados
82	—	Escrita
00 AA	—	Endereço da interface de variável de sistema
5A	D11	Dispara a operação
02	D10	Modo: escrever bloco de 32 KB na NOR
01 00	D9:D8	Bloco 0x0100 – o arquivo de ID 32
80 00	D7:D6	Área de espera em 0x8000
00 14	D5:D4	Atraso de 20 ms antes da próxima escrita
00 00 00 00	D3:D0	Não definidos

5.5. VERIFICAR O FIM DA GRAVAÇÃO

Leia a primeira word de 0x00AA e observe o byte de comando. Enquanto ele voltar 0x5A, a operação ainda está em andamento:

```
Envia: 5A A5 04 83 00 AA 01
Recebe: 5A A5 06 83 00 AA 01 5A 02 → gravando

Envia: 5A A5 04 83 00 AA 01
Recebe: 5A A5 06 83 00 AA 01 00 02 → concluída
```

O 0x5A zerado é o sinal de conclusão; o 0x02 que permanece é apenas o modo, que não é limpo.

5.6. REINICIAR

```
5A A5 07 82 00 04 55 AA 5A A5
```

Escrever 0x55AA 5AA5 na variável de sistema 0x0004 reinicia o ASIC – equivale a um ciclo de energia e faz a tela recarregar os arquivos da Flash.

6. IMPLEMENTAÇÃO NO MCU

Esqueleto do laço de atualização. A espera ativa pelo fim da gravação é a parte que não pode ser omitida: enviar o bloco seguinte antes de a Flash liberar produz arquivo corrompido sem nenhum aviso.

```
#define RAM_BUF      0x8000      // área de espera (words)
#define CHUNK        240         // bytes de dado por quadro
#define BLOCK_BYTES  (32u * 1024u)

// Envia um pedaço do arquivo para a RAM de variáveis
void dgus_write_ram(uint16_t vp, const uint8_t *data, uint8_t len) {
    uint8_t h[6] = { 0x5A, 0xA5, (uint8_t)(len + 3), 0x82,
                     (uint8_t)(vp >> 8), (uint8_t)(vp & 0xFF) };
    uart_send(h, 6);
    uart_send(data, len);
}

// Consolida na Flash o bloco já montado na RAM
void dgus_flash_block(uint16_t block, uint16_t ram, uint16_t delay_ms) {
    uint8_t f[18] = { 0x5A, 0xA5, 0x0F, 0x82, 0x00, 0xAA,
                     0x5A, 0x02,
                     (uint8_t)(block >> 8), (uint8_t)(block & 0xFF),
                     (uint8_t)(ram >> 8), (uint8_t)(ram & 0xFF),
                     (uint8_t)(delay_ms >> 8), (uint8_t)(delay_ms & 0xFF),
                     0x00, 0x00, 0x00, 0x00 };
    uart_send(f, 18);
}

// true enquanto a gravação não terminou (byte de comando ainda 0x5A)
bool dgus_flash_busy(void) {
    uint8_t q[7] = { 0x5A, 0xA5, 0x04, 0x83, 0x00, 0xAA, 0x01 };
    uart_send(q, 7);
    uint8_t r[9];
    if (!uart_recv(r, 9, 200)) return true;    // sem resposta: trate como ocupado
    return r[7] == 0x5A;
}

void atualiza_arquivo(uint8_t file_id, const uint8_t *buf, uint32_t size) {
    uint16_t block = (uint16_t)(file_id * 256u / 32u);
    uint32_t off = 0;

    while (off < size) {
        uint32_t bloco = (size - off < BLOCK_BYTES) ? (size - off) : BLOCK_BYTES;
        uint16_t vp = RAM_BUF;

        for (uint32_t i = 0; i < bloco; i += CHUNK) {
```

```
uint8_t len = (bloco - i < CHUNK) ? (uint8_t)(bloco - i) : CHUNK;
dgus_write_ram(vp, buf + off + i, len);
vp += len / 2;                // endereço em words
delay_ms(5);                  // respeite o tempo de ciclo
}

dgus_flash_block(block, RAM_BUF, 20);
while (dgus_flash_busy()) delay_ms(20);

block++;
off += bloco;
}
}
```

COMPRIMENTO ÍMPAR – o endereçamento da RAM é por word. Se o último pedaço tiver número ímpar de bytes, complete com um byte 0x00 para fechar a word; o dado excedente cai fora da região útil do arquivo.

7. DIAGNÓSTICO

Os sintomas de uma atualização malfeita são poucos e bem característicos.

SINTOMA	CAUSA PROVÁVEL	VERIFICAÇÃO
Tela preta após o reset	Bloco de destino errado — gravou por cima de outro arquivo e apagou os dados de imagem	Refaça a conta do 3.1 e confira o ID do arquivo contra o projeto no DGUS
Tela preta, destino correto	Fatiamento errado: quadro perdido, endereço fora de passo ou comprimento incorreto	Releia o bloco com o modo 0x01 e compare com o arquivo original
Imagem aparece pela metade	Comprimento do último quadro deixado em 0xF3 quando os dados eram menos de 240 bytes	Recalcule o comprimento real do último quadro
Arquivo gravado parcialmente	Bloco seguinte enviado antes de a gravação anterior terminar	Confirme que o laço espera o byte de comando zerar a cada bloco
Nada muda após o reset	Reset não executado, ou arquivo gravado em ID que o projeto não referencia	Reenvie o comando de reset; confira qual ID o controle usa
Módulo travado	Queda de energia durante a gravação	Regrave o projeto completo por cartão SD íntegro; persistindo, o caminho é a interface PGT05

ANTES DE ATUALIZAR EM CAMPO — teste a rotina completa em bancada, com o mesmo modelo de módulo e o mesmo arquivo. Garanta alimentação estável durante toda a gravação: uma queda no meio da escrita pode causar tela preta e dano à Flash.

8. INTERFACES VIZINHAS

A 0x00AA trata dos arquivos de projeto na Flash externa. Outras três interfaces cobrem casos próximos e são frequentemente confundidas com ela.

ENDEREÇO	NOME	DO QUE TRATA
0x0006	OS_Update_CMD	Grava programa DWIN OS (28 KB por vez, a partir de 0x1000) ou código 8051 (64 KB por vez) na Flash interna. É o que o DWIN OS Builder usa ao baixar pela UART2.
0x0008	NOR_FLASH_RW_CMD	Lê e grava o banco de dados NOR interno do chip – dados de usuário persistentes, não arquivos de projeto. Também exige checar o byte de comando zerar.
0x00FE	Download rápido por UART1	Escrever 0x5AA5 habilita o modo de download de alta velocidade pela UART1, usado pelas ferramentas de PC da DWIN.

Para carga inicial ou troca completa do projeto, o caminho continua sendo o cartão SD: cartão SD ou SDHC, FAT32 com setor de 4 KB, arquivos na pasta `DWIN_SET` na raiz, sem subpastas, e sem troca a quente.

FONTE

Todos os parâmetros, faixas e quadros deste material foram conferidos contra o *T5L DGUS II Application Development Guide* V2.9, capítulo 5 – interface de variável de sistema 0x00AA e exemplo de atualização de 32.ICL pelo modo 0x02.